

Documento de arquitectura

MISW 4204 – Desarrollo de software en la nube

Simón Buriticá

Jhonn Sebastian Calderón Bravo

Diego Andrés Naranjo Ríos

Juan Pablo Rodríguez García

Universidad de los Andes – 2024

Diagrama de clases:

En la figura 1 se muestra el diagrama de clases obtenido a través de la lectura expuesta por la IDRL para la competencia. Se exponen 3 clases principales las cuáles son “usuario”, que representa al piloto que quiere participar en la competición, “video” que interacciona directamente con el usuario ya que un mismo piloto puede tener ningún o muchos videos, y “task” que representan las tareas de edición de los videos. Adicionalmente, se tiene una enumeración que identifica el estado de las tareas.

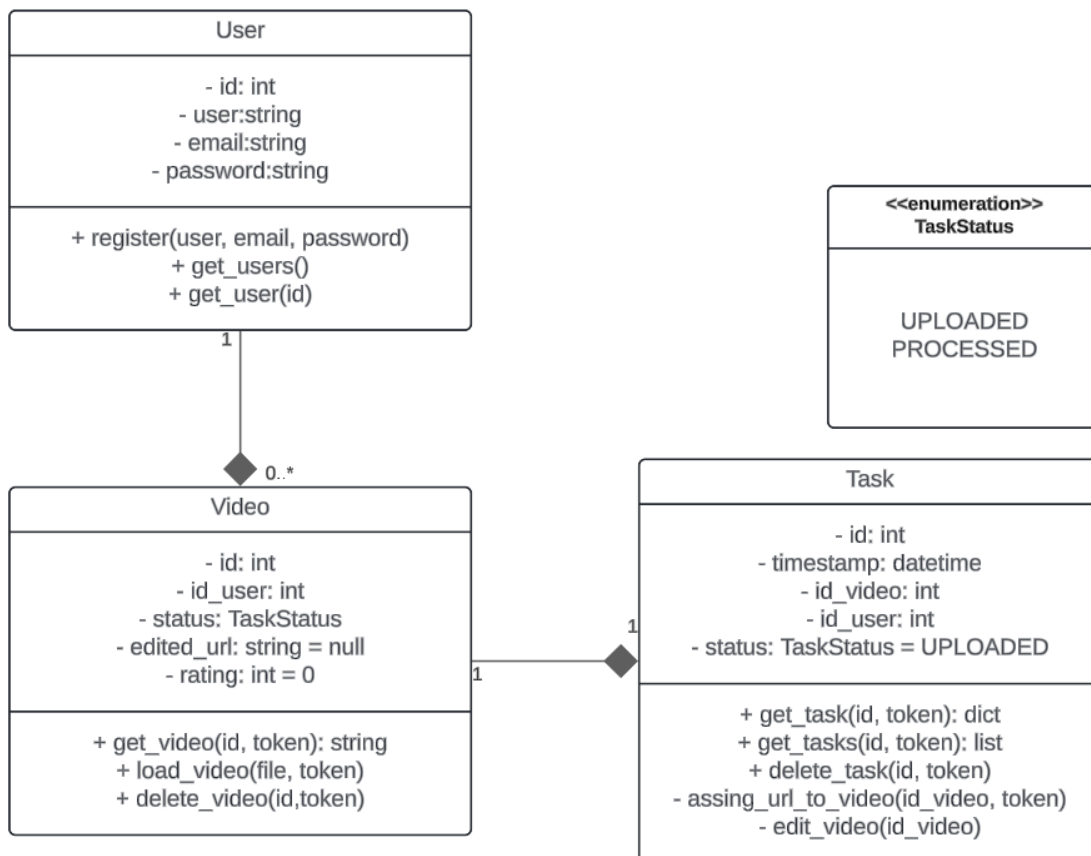


Figura 1 - Diagrama de clases preliminar a la construcción

Diagrama funcional:

Para la solución del proyecto se construyó un diagrama de componentes que tiene las siguientes características:

- Un proxy inverso que recibe los estímulos de los usuarios y se encarga de comunicarlos a la API REST. Este también cumple la función de gestionar múltiples peticiones y actuar como un balanceador de carga, al igual que enmascarar los métodos y atributos del backend, dando una capa extra de seguridad al desarrollo
- Una API Gateway que desacopla el llamado a los endpoints de cada modelo
- Un autorizador que se encarga de generar los tokens a los usuarios autenticados
- Una clase que se encarga de gestionar la información de los usuarios
- Las clases principales del modelo lógico
- Un message bróker que recibe los eventos de creación de tareas
- Un componente llamado “video worker”, el cuál se encarga de desencolar los mensajes, inicializar las tareas de edición de los videos, y enviar la petición al componente video para enlazar el video editado

En la figura 2, es posible ver el diseño propuesto, el cuál está enfocado en ser un sistema desacoplado con alta modificabilidad, disponible y seguro

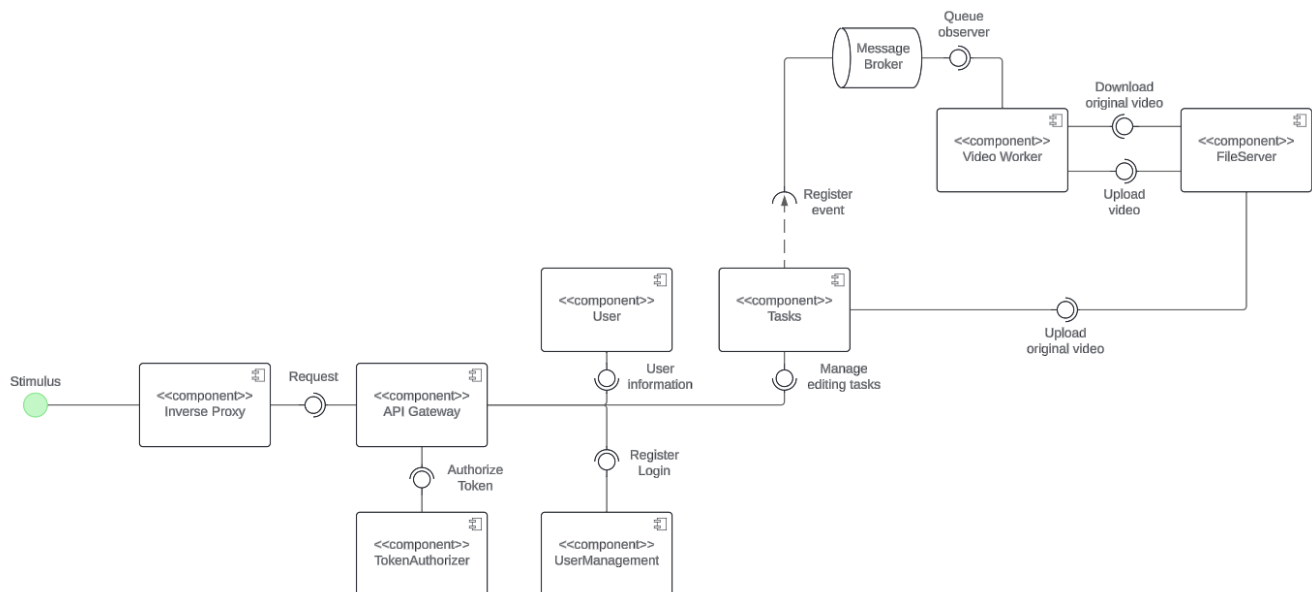


Figura 2 - Diagrama funcional para la solución del proyecto

Diagrama de despliegue

En cuanto al despliegue de la solución se pensó en construir una arquitectura basada en contenedores, dónde se exponen los diferentes servicios de forma independiente. La API se pensó como un sistema monolítico, sin embargo, acoplado a contenedores independientes para el proxy, la base de datos y el message broker. De esta manera se pensó en un sistema integrado por Docker-compose que permite un fácil despliegue y que puede ser escalado a máquinas virtuales

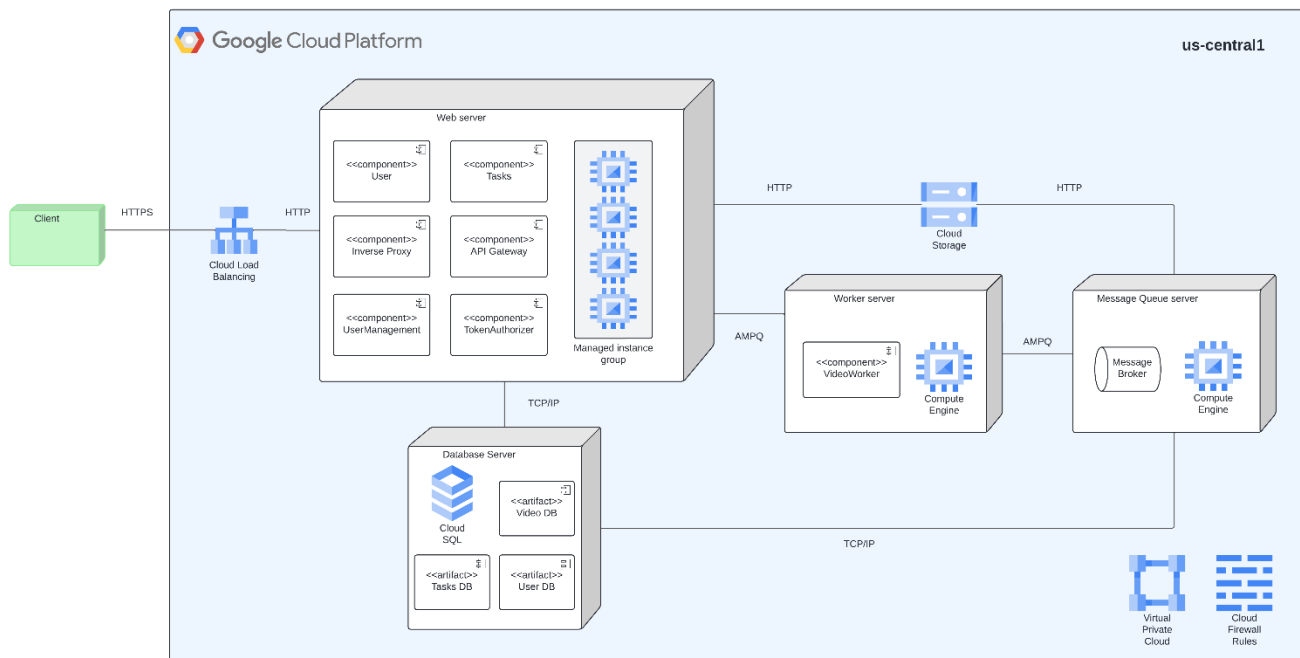


Figura 3 - Diagrama de despliegue propuesto para operar con contenedores

Iteración 3

Para la iteración 3 de la arquitectura se realizó la actualización del diagrama de despliegue para contener los servicios de Google Cloud Platform. Adicionalmente, se muestra el reemplazo del fileservidor con protocolo FTP por el servicio de Cloud Storage con almacenamiento por medio de buckets. Se agregó el servicio de load balancing que ahora actúa como el punto de ingreso para las peticiones externas del cliente. Este load balancer se comunica ahora con un grupo de instancias que contienen una regla de autoescalamiento cuándo **las instancias superan el 80% de uso de CPU**. Este grupo de instancias tiene la posibilidad de escalar hasta un **máximo de 4**, con el fin de poder garantizar un gasto adecuado de los recursos disponibles.